学校编号:10384　　分类号:_____密级:_____
学号:200223015　　UDC:_____

硕　士　学　位　论　文

Helmholtz问题谱元法并行计算

A Parallel Spectral Element Method of the Helmholtz Problem

洪　桃　李

指导教师姓名: 许传炬　教 授
申请学位级别: 硕　士　学　位
专 业 名 称: 计　算　数　学
论文提交日期: 2005 年　05 月
论文答辩日期: 2005 年　05 月
学位授予单位: 厦　门　大　学
学位授予日期: 2005　年　　月

答辩委员会主席: _____
评阅人: _____

2005 年 05 月　日

理 学 硕 士 学 位 论 文

Helmholtz问题谱元法并行计算

洪桃李

厦门大学
二〇〇五年五月

Master Dissertation

A Parallel Spectral Element Method of the Helmholtz Problem

by

Taoli Hong

Supervisor: Prof. Chuanju Xu

Speciality: Computational Mathematics

Direction: Parallel Computing

Institution: Department of Computational Mathematics
Xiamen Universtity
Xiamen, China

目 录

摘　要

本文考虑谱元法求解Helmholtz问题，利用并行计算实现了Helmholtz问题的快速求解。在一维情况下使用整体有限元预条件共扼梯度法设计了一种并行的迭代算法，通过一系列数值实验证明了这种并行算法的有效性。二维情况下，由于整体有限元预条件矩阵构造十分费时，我们选择用Schur补方法求解Helmholtz问题的谱元离散方程。在求解Schur补系统时使用重叠Schwarz区域分解构造预条件。Schur矩阵与向量的乘积以及预条件矩阵的计算都进行了并行处理。根据算法的特点，适当的在主处理器和从处理器之间进行分工，使得方法简洁容易实现，并达到了比较好的并行效果。同时通过数值实验考察了重叠区域大小以及分解子区域数量对计算量的影响。

关键词： 谱元法(SEM); Schur补(Schur Complement); 区域分解(Domain Decomposition); 预条件; 并行计算

Abstract

Spectral element method is a high-order method, and has nicer parallel feature as compared with low order methods. In this paper, a parallel preconditioned conjugate gradient iterative method is proposed to solving the spectral element approximation of the Helmholtz problem. In 1D case, the preconditioner is constructed by using the piecewise linear finite element method based on global Gauss-Lobatto points. In 2D case, the original SEM Helmholtz system is first reduced into a system corresponding to the elemental boundaries by using Schur decomposition. Then, this boundary system is preconditioned by using the additional Schwarz decomposition method. A series of numerical experiments is carried out to show that the parallel algorithm has good performance as compared to non parallel cases.

Key words: Spectral Element Method; Schur Complement; Domain Decomposition; Preconditioning; Parallel Computing;

第一章 一维Helmholtz问题谱元法并行计算

§1.1 引言

谱元法(SEM)本质上是基于高阶Lagrangian插值和利用相应数值积分逼近偏微分方程弱解的一种方法。它结合了有限元方法处理计算区域的灵活性和谱格式快速收敛的优点[7]。由于这些优点，谱元法现在已成为流行计算方法之一。谱元法的另一个好处是它便于进行并行计算。在一般应用中，由于使用高阶多项式，大量的计算是在各个元的内部进行。这个特征使谱元法并行计算切实可行。

在谱元法中[7]所采用的传统方法是在结构化的元（即四边形或六面体）中使用张量型节点基函数。这些基是基于Gauss Lobatto Legendre点的Lagrange多项式。在大区域问题中，元之间的大量交接导致刚性矩阵的稠密，分解的代价非常昂贵。因为需要大量的内存，使用直接方法是不经济的[5]。在过去的十年中，一般用迭代法求解各种方程的谱元离散问题。但由于Laplacian算子的谱元刚性矩阵的条件数是$O(KN^{d+1})$[2]，其中K是元的数量，N是多项式阶数，d是空间维数，简单的迭代法不是十分有效，因为事实上需要太多的迭代次数才能达到收敛。长期以来，人们尝试用各种预条件来解决这个问题。Orszag [10]和Deville et al. [3]分别采用有限差分和有限元模型作为谱矩阵的预条件。

本章根据Orszag和Deville等人的思想，讨论了有限元预条件谱元系统的快速求解方法。精确地讲，我们考虑并行的预条件共扼梯度法，用线性有限元方法作为谱元刚性矩阵的预条件。我们的目的是进行一系列的数值实验来表明用并行算法可以有效地减少求解Helmholtz问题谱元系统的CPU时间。

§1.2 问题简述

考虑以下一维Helmholtz问题: 找一个定义在$\Omega = (a,b)$上的函数u, 使得

$$\begin{cases} -(pu')' + \lambda^2 u = f, \text{ in } \Omega \\ u(a) = u(b) = 0 \end{cases} \tag{1.1}$$

这里λ 是一个实数, ' 表示关于x 的导数, $f(x), p(x)$ 是定义在Ω上的函数,假设存在两个正常数τ_0 和τ_∞ 使得

$$\tau_\infty > p(x) > \tau_0, \forall x \in \Omega. \tag{1.2}$$

(1.1)问题的变分形式是: 找$u \in H_0^1(\Omega)$，使得

$$\int_\Omega p(x)u'(x)v'(x)dx + \lambda^2 \int_\Omega u(x)v(x)dx = \int_\Omega f(x)v(x)dx, \forall v \in H_0^1(\Omega) \tag{1.3}$$

谱元离散首先需要把区间Ω 分解成区域元（谱元）

$$\Omega = \bigcup_{k=1}^{K} \Omega_k$$

这里设第k个谱元的长度是l_k。逼近解u_h的谱元空间X_h是$H_0^1(\Omega)$的子空间，它是所有阶数小于或等于N的分片多项式集合:

$$X_h = H_0^1(\Omega) \cap P_{N,K}(\Omega)$$

这里

$$P_{N,K}(\Omega) = \{\phi \in L^2(\Omega), \phi_{|\Omega_k} \in P_N(\Omega_k), k = 1, \cdots, K\}.$$

考虑谱元离散问题: 找$u_h \in X_h$ ，使得

$$\int_\Omega p(x)u_h'(x)v_h'(x)dx + \lambda^2 \int_\Omega u_h(x)v_h(x)dx = \int_\Omega f(x)v_h(x)dx, \forall v_h \in X_h \tag{1.4}$$

让$\xi_0, \xi_1, \cdots, \xi_N$ 表示Gauss-Labatto-Legendre (GLL) 点, 即ξ_i满足:

$$\xi_0 = -1, \xi_N = 1, L_N'(\xi_i) = 0, \forall i \in \{1, \cdots, N-1\}.$$

相应的用于GLL数值积分公式中的权系数表示为ρ_i。

令$\Omega_k = [a_{k-1}, a_k]$, 那么全局GLL点和相应的权系数定义如下：

$$\xi_{i,k} = a_{k-1} + (\xi_i + 1)l_k/2, \ \rho_{i,k} = \rho_i l_k/2, \ \forall i, \leq i \leq N, \ \forall k, 1 \leq k \leq K.$$

使用著名的Gauss-Lobatto 数值积分逼近问题(1.4)中的积分，得到下面的谱元离散问题: 找$u_h \in X_h$ 使得

$$\begin{aligned}&\sum_{k=1}^{K}\sum_{i=0}^{N} p(\xi_{i,k})u_h'(\xi_{i,k})v_h'(\xi_{i,k})\rho_{i,k} + \lambda^2 \sum_{k=1}^{K}\sum_{i=0}^{N} u_h(\xi_{i,k})v_h(\xi_{i,k})\rho_{i,k} \\ &= \sum_{k=1}^{K}\sum_{i=0}^{N} f(\xi_{i,k})v_h(\xi_{i,k})\rho_{i,k}, \qquad\qquad \forall v_h \in X_h.\end{aligned} \tag{1.5}$$

为了得到矩阵形式，我们必须为离散空间X_h 选择一组基。在众多可选择的基中，最自然的选择是基于单元GLL点的Lagrangian插值函数[7]。

对于任意一个$w_h \in X_h$ ，将w_h表示为

$$w_h^k(r) = \sum_{i=0}^{N} w_i^k h_i(r),\ x \in \Lambda_k \to r \in \Lambda \tag{1.6}$$

这里h_i满足：

$$h_i \in P_N(\Lambda), h_i(\xi_j) = \delta_{ij}, \forall i,j \in \{0,\cdots,N\} \tag{1.7}$$

而$w_i^k = w_h(\xi_{i,k})$. w_h 在交界上的连续性和边界条件由下列关系给定：

$$w_N^k = w_0^{k+1},\ \forall k \in \{1,\cdots,K-1\}$$

$$w_0^1 = w_N^K = 0.$$

把u_h 通过这种形式表达，并且选择每个仅在一个全局GLL点上不为零的函数v_h做为测试函数，我们得到以下矩阵形式的方程：

$$\sum_{k=1}^{K}{}' \sum_{j=0}^{N} S_{ij}^k u_j^k = \sum_{k=1}^{K}{}' \sum_{j=0}^{N} B_{ij}^k f(\xi_{j,k}) \tag{1.8}$$

这里

$$\begin{cases} S_{ij}^k = L_{ij}^k + \lambda^2 B_{ij}^k \cdots\cdots\cdots\cdots \forall i,j \in \{0,\cdots,N\} \\ L_{ij}^k = \frac{4}{l_k^2}\sum_{q=0}^{N} D_{qi}D_{qj}p(\xi_{q,k})\rho_{q,k} \cdots\cdots\cdots \forall i,j \in \{0,\cdots,N\} \\ B_{ij}^k = \rho_{i,k}\delta_{ij} \cdots\cdots\cdots\cdots\cdots\cdots \forall i,j \in \{0,\cdots,N\} \end{cases}$$

这里$D = (D_{ij})_{(N+1)\times(N+1)}$ 表示Legendre 导数矩阵[1], Σ' 表示保证了u_h在各个元交界上连续和边界上满足边条件的“刚性和”。

§1.3 有限元预条件

因为谱元离散矩阵的条件数通常很大，为使迭代有效必须使用预条件。在这一节中，我们使用Deville [3]提出的基于GLL网格点的线性

有限元预条件。Parter [8] 和黄等人[6]研究和发展了这个预条件方法。简要的说, 有限元矩阵P有和谱元矩阵S 相似的结构：

$$S = \begin{pmatrix} S^1 & & & \\ & \oplus & & \\ & & \ddots & \\ & & & \oplus \\ & & & & S^K \end{pmatrix} \quad P = \begin{pmatrix} P^1 & & & \\ & \oplus & & \\ & & \ddots & \\ & & & \oplus \\ & & & & P^K \end{pmatrix}$$

这里符号$\oplus$ 定义如下：如果B，C 是如下形式两个矩阵，

$$B = \begin{pmatrix} \hat{B} & \alpha \\ \alpha^T & b \end{pmatrix} \quad C = \begin{pmatrix} c & \beta^T \\ \beta & \hat{C} \end{pmatrix}$$

那么

$$\begin{pmatrix} B & \\ & \oplus \\ & C \end{pmatrix} = \begin{pmatrix} \hat{B} & \alpha & \\ \alpha^T & b+c & \beta^T \\ & \beta & \hat{C} \end{pmatrix}$$

矩阵S^1, S^K, P^1, P^K 是$N \times N$ 矩阵，而$S^k, P^k, 2 \le k \le K-1$，是$(N+1) \times (N+1)$ 矩阵。S^k 和P^k之间的一个区别是S^k 是满的而P^k 是三对角的。黄和许[6] 已经证明对矩阵S 使用预条件P 后的条件数与N 以及K无关。

§1.4 并行预条件共扼梯度法

1.4.1 MPI 介绍

在我们的数值算法设计中使用*Message-Passing Interface* (MPI)，它是一个标准的并行计算环境。使用MPI基于以下原因：第一，MPI是一个消息传递库的标准规范，它的实现MPICH已经广泛用于并行和分步式计算环境中。第二，MPICH是免费的，它支持许多操作系统，包括Windows, Linux, Unix,以及许多编程语言，包括Fortran, C, C++。第三，基于MPI的程序既可在一台拥有很多CPU的计算机上运行，也可以在工作站集群中运行。

1.4.2 并行算法

现在我们来详细描述谱元Helmholtz 系统并行算法。作为一种强大

的迭代方法，PCG方法在线性系统求解，特别是在对称正定系统求解中应用最广泛的方法之一[9]。一般说来，在PCG每次迭代当中我们需要求解预条件一次，并且计算矩阵向量积一次。在整个计算当中，谱元刚性矩阵的形成和矩阵向量积计算花费最多。

假设我们有$M(M>1)$个处理器，我们称第一个处理器为主处理器，其它$M-1$个处理器为从处理器。一个处理器可以是一个CPU或集群中的一个工作站。我们按照以下方法来给每个处理器分配任务。

首先我们把K 个元分成M 组: $K_1,\cdots,K_M$, 使得

$$\sum_{n=1}^{M} K_m = K,\ K_m \geq 0, m=1,\cdots,M,$$

然后我们把每个组分配给一个处理器。第m个处理器的任务组K_m 从k_m到k_m+K_m-1编号。因此

$$\begin{aligned} &k_m + K_m = k_{m+1},\ 1 \leq m \leq M-1 \\ &k_1 = 1 \\ &k_M + K_M = K+1 \end{aligned}$$

令$E_m := \{k_m,\cdots,k_m+K_m-1\}$, $1\leq m\leq M$，所有矩阵$S^k, k\in E_m$与向量的积通过第m 个处理器计算. 如果所有的处理器有同样的性能，每个处理器的负载应该是均衡的。在这个例子中我们需要以下的均衡条件：$|K_i-K_j|\leq 1, 1\leq, i,j\leq M$，并且

$$K_m = \left[\frac{K}{M}\right],\quad 1\leq m\leq M-(K \bmod M)$$

$$K_m = \left[\frac{K}{M}\right]+1,\quad M-(K \bmod M)<m\leq M$$

表一是一个选取K_m的例子, 这里有17个元($K=17$), 5个处理器($M=5$). 在这个例子中每个处理器被分配3或4个元。

m	E_m
1	1,2,3
2	4,5,6
3	7,8,9
4	10,11,12,13
5	14,15,16,17

表1. $K=17$ ， $M=5$时的任务分配.

在每次迭代中，第m个处理器计算$S^k u^k, \forall k \in E_m$。一旦得到所有的$S^k u^k$，主处理器组装来自从处理器的积以得到刚性和。其它相关的工作比如全局范数计算，判断是否收敛等等，由主处理器来完成。

总的来说，Helmholtz方程的预条件谱元算法并行过程可以表述如下：

(1) 主处理器计算有限元预条件。

(2) 每个处理器计算相应的$S^k u^k, \forall k \in E_m$ 。

(3) 每个从处理器发送向量中各个元对应的部分到主处理器，主处理器组装成一个完整的向量。得到“刚性和”。

(4) 主处理器求解全局预条件，以及其他一些PCG方法需要的工作。

(5) 主处理器计算余量，决定线性系统的解是否已经达到需要的精度，如果没有，主处理器发送向量中相应的部分给各个处理器，然后回到第(2) 步。

以上过程见图(1.1)所示。

注 1: 与每个处理器中的矩阵向量积计算相比，不同的处理器之间的数据交换相对较小。实际上预条件是由主处理器通过直接方法来求解的，不需要处理器之间的通信。从下面出数值实验可知我们的算法需要的数据交换是很理想的。

注 2: 在第(2)步中,为了减少矩阵向量积的CPU时间，可以显式地构造谱元矩阵。然而二维和三维情况下，谱元矩阵的存储需要大得多的内存（二维时为$O(KN^4)$，三维时为$O(KN^6)$），当K或N变大时，这是不现实的。因此，隐式计算矩阵向量积是不可避免的。这种变化会节省内存，但花费更多时间。

§1.5 数值实验

在这一节，我们报告数值实验的结果。所有的测试工作是在中科

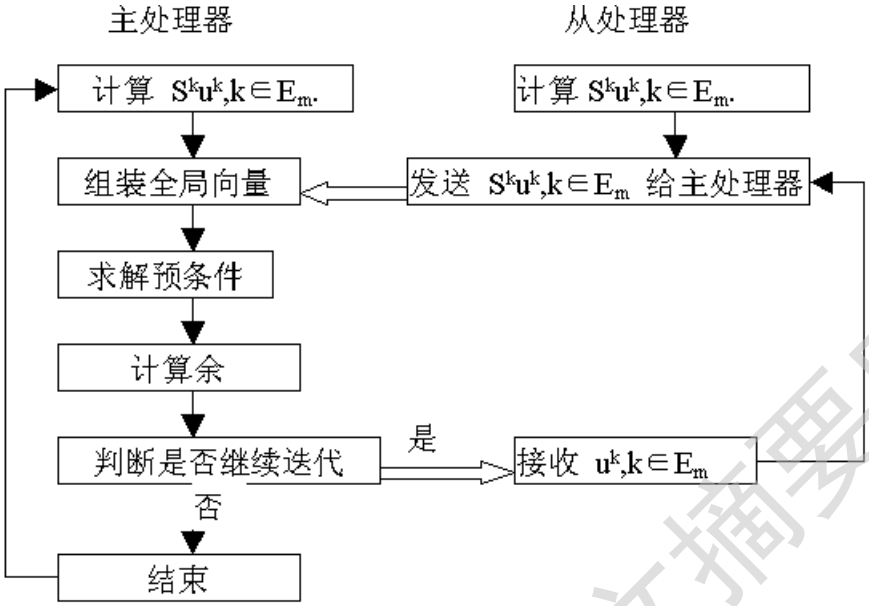

图 1.1 并行PCG算法示意图.

院科学和工程计算国家重点实验室(LSEC)的SGI Origin3800服务器上完成的。

数值计算参数选择如下：

$$\Omega = (0, 2\pi)$$
$$p(x) = \delta$$
$$\lambda = 1$$
$$f(x) = (\delta k^2 + 1)\sin(kx).$$

在上述设置下, 问题(1.1) 的精确解是已知的：

$$u(x) = \sin(kx)$$

在所有的测试中, 我们选择$\delta = 0.001, k = 100$, 记录时间单位是10毫秒. 区域被划分成相等大小的元。

测试 1. 随着K变化，PCG收敛研究以及各个计算步骤花费CPU时间的报告。

算法参数：$N = 3, M = 1$, 谱元矩阵在内存中存储，使用预条件。

K	1000	2000	4000	8000	16000
迭代次数	5	6	6	5	5
最大误差	4.78e-06	1.51e-07	4.75e-09	1.48e-10	5.72e-12
总时间	16	33	67	128	258
创建预条件	11	23	45	90	180
创建矩阵S	1	2	4	8	16
PCG 运算	3	7	14	23	48

表2. $N = 3, M = 1$ 时的PCG收敛情况，误差和CPU时间。

从表2中，我们可以看到当N固定时，随着K的增长，CPU时间近似呈线性增长，这与我们的算法设计是一致的。同时可以看到谱元解的误差与元的个数的函数关系。和预期的一样，误差阶是$O(h^{N+1})$，这里h是元的大小。另外迭代次数与K无关，进一步说明了有限元预条件的有效性。

测试 2. 和测试1类似, 但K 固定，N在变化。

$K = 500, M = 1$, 谱元矩阵在内存中存储，使用预条件。

N	3	5	7	9	11
迭代次数	3	5	7	9	11
最大误差	1.55e-4	1.85e-7	1.67e-10	1.78e-13	1.03e-13
创建矩阵S	1	2	3	5	9
PCG 运算	2	3	5	9	14

表3. $K = 500, M = 1$时的PCG收敛情况，误差和CPU时间。

在表3中我们进行了与表2类似的比较，但这里K是固定的。我们发现，随着N在一个小范围内增大，迭代次数随着N线性增加。然而当N比较大时，迭代次数似乎稳定下来，如图1.2所示。出现这种现象的原因尚不明朗。从表3可以很清楚地观察到随N增大的谱收敛情况。这和谱元法的一般理论相吻合。

测试 3. 并行计算: 比较处理器增加时的CPU时间变化。

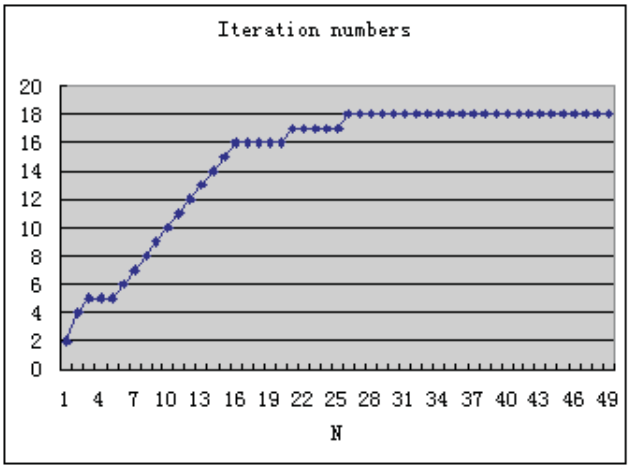

图 1.2 迭代次数随多项式阶数N 的变化关系

算法参数：$K = 500, N = 11$, 谱元矩阵在内存中存储，使用预条件。

M	1	2	4	8
创建矩阵S	8	4	2	1
PCG运算	14	10	8	7

表4. 使用不同数量处理器所需要的CPU时间.

在表4中，我们列出了使用不用数量处理器时花费CPU的时间。当处理器数量M增加时，构造谱元矩阵S的时间呈线性下降，但是PCG迭代所需时间下降的比较慢。这是因为求解预条件的过程不是并行的，并且随着M的增加，通信的时间在增加。

测试 4: 和测试3类似，但是谱元矩阵S 没有在内存中存储.

算法参数：$K = 500, N = 11$, 矩阵S没有存储，使用预条件。

M	1	2	4	8
总时间	89	51	25	16

表5. 和表4 类似, 但矩阵S没有在内存中存储。

我们重复了测试3，但矩阵S没有在内存中存储。在这种情况下，矩阵向量积的计算时间更多，因而预计并行会更加有效。在表5中，

列出了计算过程中使用不同数量处理器所用的总CPU时间。我们观察到当处理器数量M增加时，总CPU时间迅速减少（几乎随着M线性减少）。